Prefix Codes for Power Laws with Countable Support

Michael B. Baer, *Member, IEEE*

***Abstract*— In prefix coding over an infinite alphabet, methods that consider specific distributions generally consider those that decline more quickly than a power law (e.g., Golomb coding). Particular power-law distributions, however, model many random variables encountered in practice. For such random variables, compression performance is judged via estimates of expected bits per input symbol. This correspondence introduces a family of prefix codes with an eye towards near-optimal coding of known distributions. Compression performance is precisely estimated for well-known probability distributions using these codes and using previously known prefix codes. One application of these near-optimal codes is an improved representation of rational numbers.**

***Index Terms*— Coding of integers, continued fractions, infinite alphabet, optimal prefix code, power law, rational numbers, search trees, Shannon entropy.**

I. Introduction

Consider discrete power-law distributions, those of the form

$$p(i) \sim ci^{-\alpha}$$

for constants $c > 0$ and $\alpha > 1$, where $p(i)$ is the probability of symbol i, and $f(i) \sim g(i)$ implies that the ratio of the two functions goes to 1 with increasing i. Such distributions could be either inherently discrete or discretized versions of continuous power-law distributions.

Several researchers in varied fields have, in classic papers ranging from decades to centuries old, observed power-law behavior for various discrete phenomena. These include distribution of wealth [1], [2], town and city populations [2], [3], word frequency [2], [4], [5], numbers of species of a given genus [2], [6], and terms in continued fractions [7], [8]. More recent papers model various Internet phenomena [9]. So active is the topic that several surveys and popular expositions exist, e.g., [9]–[11].

However, there has been relatively little work on lossless compression of symbols obeying such distributions, in spite of a rich literature on prefix coding problems [12]. Exponential-Golomb codes [13] (generalizations of Elias' γ code [14]) are a good fit for certain power laws [15], [16], leading to their widespread use in compressing video and numerical data [15], [17]. To the author's knowledge, though, only one specific infinite-cardinality power-law distribution, the Gauss-Kuzmin distribution [18, p. 341], has been used to judge compression performance of prefix codes [19], [20].

Here we propose simple codes which not only improve upon existing codes for encoding symbols distributed according to the Gauss-Kuzmin distribution — which applies to coding rational numbers using continued fractions — but also efficiently code other common distributions, such as the zeta distribution with parameter 2 [21], [22]. We estimate compression performance for dozens of code/distribution combinations. For fixed codes, these estimates are rigorously shown to be precise.

II. Background, formalization, and motivation

The most common infinite-alphabet codes are codes that are optimal for geometric [23], [24] and geometrically-based [25]–[29] distributions. For geometric distributions, these are known as Golomb codes, and are based on the *unary code* — ones terminated by a zero, i.e., a code consisting of codewords the form $\{1^j0\}$ for $j \geq 0$. In a Golomb code (Gk), a unary code prefix precedes a binary code suffix. This binary suffix is a *complete binary code*, in that it has (k) codewords of the same length or length differing by at most one. For example, the alphabetic complete binary code of size three that is monotonically nonincreasing in length is $\{0, 10, 11\}$, so the Golomb code G3 is $\{00, 010, 011, 100, 1010, \ldots\}$. If the complete binary code suffix is of constant length, the overall Golomb code is also called a Rice code. Rice codes are used in standards such as JPEG-LS [30]. Codes that exhibit an efficient coding rate for power laws, by contrast, are not known to be optimal (excepting those with finite support and trivial examples for dyadic probability mass functions).

We restrict ourselves to binary codes and assume that the symbols to be coded are positive integers. Thus, an infinite-alphabet source emits symbols drawn from the alphabet $\mathcal{X} = \{1, 2, 3, \ldots\}$. (Some applications code the alphabet $\mathcal{X}_0 = \{0, 1, 2, \ldots\}$ or the alphabet $X_{\mathbb{Z}} = \{0, -1, 1, -2, 2, \ldots\}$, but any code of either form can be mapped trivially to a code on $\mathcal{X}$.) Symbol i has probability $p(i) > 0$, forming probability mass function $P = \{p(i)\}$. The source symbols are coded into binary codewords. The codeword $c(i) \in \{0, 1\}^*$, corresponding to symbol i, has length $n(i) \in \mathbb{Z}_+$, thus defining length distribution $N = \{n(i)\}$. An optimal code is one that minimizes $\sum_{i\in\mathcal{X}} p(i)n(i)$ with the constraint of a corresponding code being uniquely decodable, which one is if and only if the Kraft inequality, $\sum_{i\in\mathcal{X}} 2^{-n(i)} \leq 1$, is satisfied. We can assume without loss of generality that these codes are prefix codes, that is, codes where there are no two codewords of the form $c(i)$ and $c(j) = c(i)x$, where $c(i)x$ denotes the concatenation of strings $c(i)$ and (nontrivial) x. (In a similar use of notation, 0^k and 1^k denote k 0's and k 1's, respectively. Note also that we use $\lg$ to denote $\log_2$ and $\ln$ to denote $\log_e$, where e is the base of the natural logarithm.)

One cannot use the Huffman source coding algorithm [31] to find an optimal code, as one can for a finite source alphabet. However, it is sensible that a code over the integers should be *monotonic*, that is, that $n(i) \geq n(i+1)$ for all $i \geq 0$. An exchange argument easily shows that this is necessary for the code to be optimal given a distribution for which $p(i) > p(i+1)$ for all i.

Also desirable is for a code to be *alphabetic* or *order preserving*; that is, if $c(i, j)$ is the jth bit of the ith codeword, then $c(i+1, j) < c(i, j)$ only if there is a $k < j$ such that $c(i+1, k) \neq c(i, k)$. Alphabetic codes allow the prefix coding tree to be used as a decision tree, which is useful for search problems, as in [32], [33]. It is also useful for implementation of arithmetic coding: Because binary arithmetic coding is much faster than other types of arithmetic coding, a decision tree can reduce an infinite-alphabet source into a binary source for fast arithmetic coding, as in [15]. In addition, order preservation is necessary for the ordered representation of rational numbers as integers in continued fractions [19], [20]; in this correspondence we improve upon these representations.

Any valid monotonic prefix code has a (possibly different) alphabetic prefix code with the same length distribution. For example, the Elias γ code was first presented in a nonalphabetic version, then transformed into alphabetic form (as a decision tree) in [32]. Where there is ambiguity, we will assume use of the alphabetic version of a code.

Another desirable property is one we call "smoothness":

Definition: We call $N = \{n(i)\}$ *j-smooth* if, for every $i > j$, if $n(i+1) = n(i+2)$, then $n(i+1) - n(i) \leq 1$, that is, there are no "jumps" followed by "plateaus"; *weakly smooth* means that it is j-smooth for some j. Thus, for any j, a j-smooth code includes all weakly smooth codes. Similarly, 0-smooth (or *strongly smooth*) codes include all j-smooth (and thus weakly smooth) codes. Also, we call a

M. B. Baer is with Electronics for Imaging, 303 Velocity Way, Foster City, CA 94404 USA (e-mail: Michael.Baer@efi.com).

$P = \{p(i)\}$ *j-antiunary* if, for every $i > j$, $p(i) < p(i+1)+p(i+2)$; *antiunary* means that it is j-antiunary for some j.

Observation: No j-antiunary distribution has an optimal code which is not j-smooth. Thus no antiunary distribution has an optimal code which is not weakly smooth.

Proof: Suppose a j-antiunary distribution P has an optimal code with lengths N which is not j-smooth. Then there exists an $i > j$ such that $n(i+1) = n(i+2)$ and $n(i+1)-n(i) > 1$. Consider $N' = \{n'(i)\}$ for which $n'(k) = n(k)$ except at values $n'(i) = n(i) + 1$, $n'(i+1) = n(i+1) - 1$, and $n'(i+2) = n(i+2) - 1$. Clearly N' satisfies the Kraft inequality and $\sum_i p(i)n'(i) < \sum_i p(i)n(i)$, so N is not optimal. ∎

Every power law is antiunary, but most previously proposed codes suitable for power-law distributions are not weakly smooth, so they could not be optimal solutions. The proof shows that, when such codes are applied to antiunary distributions, it is always a simple matter to improve such a code for use with such a distribution.

For many probability distributions, however, there is no guarantee that an optimal code would be computationally tractable, let alone computationally practical for compression applications. We thus judge performance of candidate codes by expected bits per coded symbol rather than by strict optimality. One of the contributions of this correspondence is a comparison of various codes for well-known power-law distributions.

III. A new family of codes for integers

We propose a family of monotonic, alphabetic, computational efficient, 0-smooth codes, starting with the code shown in the center set of columns ($n_0(\cdot)$ and $c_0(\cdot)$) of Table I, which is defined as

$$c_0(i) = \begin{cases} 0b(i-1,3), & i < 4 \\ 1c_0\left(\frac{i-2}{2}\right)0, & i = \{4,6,8,\ldots\} \\ 1c_0\left(\frac{i-3}{2}\right)1, & i = \{5,7,9,\ldots\}. \end{cases}$$

The term $b(j,k)$ denotes the $(j+1)$th codeword of a complete binary code with k items, which is order-preserving (alphabetic), with the first $2^{\lceil \lg k \rceil} - k$ items having length $\lfloor \lg k \rfloor$ and the last $2k - 2^{\lceil \lg k \rceil}$ items having length $\lceil \lg k \rceil$. In this case, that means that $c_0(1) = 0b(0,3) = 00$, $c_0(2) = 0b(1,3) = 010$, and $c_0(3) = 0b(2,3) = 011$. Thus, for example, $c_0(12) = 1c_0(5)0 = 11c_0(1)10 = 110010$. This is a unary code of length m, followed by a binary digit b (where $b = 0$ or $b = 1$), and a binary code of length $m + b - 1$, and is thus straightforward to encode, decode, and write in the form of an implicit infinite search tree.

This code, like exponential-Golomb codes, is a modification of the γ code. Whereas the γ code has an m-bit unary code followed by a complete binary code for 2^{m-1} items, Code 0 follows the unary prefix by a complete binary suffix for $3 \cdot 2^{m-1}$ items. Straightforward extensions of this can be obtained by modifying the search tree. We can add a k-bit binary number to each possible codeword — as in the fourth and fifth set of columns in Table I — extending Code 0 in the same manner that Rice codes extend unary codes, that is,

$$c_k(i) = c_0\left(1 + \left\lfloor \frac{i-1}{2^k} \right\rfloor\right) b((i-1) \bmod 2^k, 2^k)$$

where $k > 0$ and $b\left((i-1) \bmod 2^k, 2^k\right)$ is the k-bit representation of $(i-1) \bmod 2^k$. Call any of the new extensions Code k.

Another extension, similar to [15] and [34], involves first coding with a finite code tree, then, if this initial codeword is all 1's, adding Code 0. If we start as in a unary code and switch to Code 0 after κ ones, then let Code $-\kappa$ denote the implied code, e.g., Code -1, the second set of columns ($n_{-1}(\cdot)$ and $c_{-1}(\cdot)$) in Table I. Formally, for $k = -\kappa < 0$,

$$c_k(i) = \begin{cases} 1^{i-1}0, & i \le -k \\ 1^{(-k)}c_0(i+k), & i > k. \end{cases}$$

All codes presented here are 0-smooth (strongly smooth), and can be coded and decoded using only additions, subtractions, and shifts such that the total number of operations is proportional to the number of encoded output bits.

IV. Application

Table II lists various distributions for which no optimal code is known and estimates, in expected bits per input symbol, of coding performance using several different codes. The entropy and the expected bits per symbol of an optimal code are also estimated. H denotes the entropy of the distribution ($H(P) = -\sum_i p(i) \lg p(i)$) and N^* (the expected codeword length of) the optimal code. Golin denotes the best Golin code [35]; Code k denotes the best of the codes introduced here; л denotes the Levenshtein (Левенштейн) code [36]; $\gamma/\delta/\omega$/EGk denotes the best of the Elias codes [14] and exponential-Golomb codes [13], which in these examples is always the Elias γ code (EG0); Y denotes Yokoo's code for the Gauss-Kuzmin distribution [20]; and Gk denotes the best Golomb code (with parameter k) [23]. These codes are defined in the cited papers and the definitions are repeated in the Appendix, which also explains the methods by which the estimations of bits per symbol are calculated. In cases for which there are multiple codes and/or parameters, the best one is chosen and indicated in superscript. Note that, as in previous papers on these and similar codes [13], [37], the best code is chosen by its empirical performance; there appears to be no simple rule for deciding which code to use.

We show the performance for the overall best fixed code for each distribution in bold in Table II, and, if a Golin code is better, this is in italics. Note that Golin codes do well for inputs with rapidly declining probabilities, whereas Yokoo's code and the codes introduced here have the best results for inverse square probability mass functions. However, Golin codes, in being calculated on the fly, are often impractical, both due to the potential for rounding errors to lead to coding errors and due to the computational complexity of the required floating point divisions.

We find that Code -1 is of particular interest as it happens to be an excellent code for the Gauss-Kuzmin distribution, defined and well-approximated as follows:

$$p^{\mathrm{GK}}(i) \triangleq -\lg\left[1 - \frac{1}{(i+1)^2}\right] \approx \frac{\lg e}{(i+1)^2}$$

This shows how it is a power law. The Gauss-Kuzmin distribution is the one for which to code when expressing coefficients of continued fractions, as in [19], [38], in which EG0 is proposed for use, and [20], in which Yokoo's code is proposed. Code -1 is only about 0.008% worse than the (approximated) optimal code, whereas Yokoo's code is 0.449% worse and the Elias γ code (EG0) is 1.007% worse.

Note also that Code -2 is a good code for the zeta distribution with parameter $s = 2$, where the zeta distribution is defined as

$$p_s^{\zeta}(i) \triangleq \frac{1}{i^s \zeta(s)}$$

and ζ is the Riemann zeta function $\zeta(s) \triangleq \sum_{i=1}^{\infty} i^{-s}$ for $s > 1$. The zeta distribution is used to model several phenomena including language [5]. Optimal codes for the zeta distribution ($s = 2$) were considered in Kato's unpublished manuscript [22]. In this work, the optimal codeword lengths for the first ten symbols are shown to lie in ranges of two possible values for each codeword (or one for the first, which has $n(1) = 1$). The codeword lengths of Code -2 all lie

	Code -2		Code -1		Code 0		Code 1		Code 2	
i	$n_{-2}(i)$	$c_{-2}(i)$	$n_{-1}(i)$	$c_{-1}(i)$	$n_0(i)$	$c_0(i)$	$n_1(i)$	$c_1(i)$	$n_2(i)$	$c_2(i)$
1	1	0	1	0	2	0 0	3	0 0 0	4	0 0 00
2	2	10	3	1 0 0	3	0 1 0	3	0 0 1	4	0 0 01
3	4	11 0 0	4	1 0 1 0	3	0 1 1	4	0 1 0 0	4	0 0 10
4	5	11 0 1 0	4	1 0 1 1	4	10 0 0	4	0 1 0 1	4	0 0 11
5	5	11 0 1 1	5	1 10 0 0	4	10 0 1	4	0 1 1 0	5	0 1 0 00
6	6	11 10 0 0	5	1 10 0 1	5	10 1 00	4	0 1 1 1	5	0 1 0 01
7	6	11 10 0 1	6	1 10 1 00	5	10 1 01	5	10 0 0 0	5	0 1 0 10
8	7	11 10 1 00	6	1 10 1 01	5	10 1 10	5	10 0 0 1	5	0 1 0 11
9	7	11 10 1 01	6	1 10 1 10	5	10 1 11	5	10 0 1 0	5	0 1 1 00
10	7	11 10 1 10	6	1 10 1 11	6	110 0 00	5	10 0 1 1	5	0 1 1 01

TABLE I
FIVE OF THE CODES INTRODUCED HERE

		H	N^*	Golin	Code k	Л	$\gamma/\delta/\omega$/EGk	Y	Gk
Gauss-Kuzmin		3.43253	3.47207	$3.50705^{(1,2)}$	$\mathbf{3.472346}^{(-1)}$	3.77915	$3.50705^{(\gamma)}$	3.48765	$\infty^{(\forall k)}$
Yule-Simon	$\rho = 1$	2.95215	2.98136	$3.^{(1,2)}$	$2.983338^{(-1)}$	3.17826	$3.^{(\gamma)}$	$\mathbf{2.98138}$	$\infty^{(\forall k)}$
	$\rho = 1.5$	2.17073	2.21571	$2.22507^{(1)}$	$\mathbf{2.230792}^{(-2)}$	2.32233	$2.28020^{(\gamma)}$	2.26031	$2.85003^{(3)}$
	$\rho = 2$	1.74685	1.83787	$1.84024^{(1)}$	$\mathbf{1.848484}^{(-4)}$	1.91747	$1.94200^{(\gamma)}$	1.92361	$2.^{(1)}$
	$\rho = 2.5$	1.47629	1.62102	$1.62191^{(1)}$	$\mathbf{1.626668}^{(-5)}$	1.68947	$1.74664^{(\gamma)}$	1.73044	$2.66666\ldots^{(1)}$
	$\rho = 3$	1.28665	1.48534	$1.48563^{(1,2)}$	$\mathbf{1.488172}^{(-6)}$	1.54608	$1.61950^{(\gamma)}$	1.60550	$1.5^{(1)}$
zeta	$s = 2$	2.36259	2.41766	$2.43310^{(1)}$	$\mathbf{2.417772}^{(-2)}$	2.53468	$2.44631^{(\gamma)}$	2.43042	$\infty^{(\forall k)}$
	$s = 2.5$	1.46525	1.65431	$1.65767^{(1)}$	$\mathbf{1.658015}^{(-4)}$	1.70907	$1.73223^{(\gamma)}$	1.71963	$1.94737\ldots^{(1)}$
	$s = 3$	0.97887	1.33453	$1.33504^{(1)}$	$\mathbf{1.336680}^{(-4)}$	1.36956	$1.42207^{(\gamma)}$	1.41389	$1.36843\ldots^{(1)}$
		entropy	"designer" codes			f i x e d c o d e s			

TABLE II
COMPRESSION (IN BITS PER SYMBOL) AND CODE PARAMETER (WHERE APPLICABLE)

within the allowed ranges. However, we can empirically find better codes, showing that Code -2, although the best simply described code we know of, is about 0.005% worse than an optimal code.

A third distribution family is that of Yule [6] and Simon [2],

$$p_\rho^{\mathrm{YS}}(i) \triangleq \rho B(i, \rho+1) \qquad \left(p_\rho^{\mathrm{YS}}(i) = \rho\frac{(i-1)!\rho!}{(\rho+i)!}\right)$$

where $B(i,j)$ is the beta function, $\rho > 0$, and the right equation applies for integer ρ. Thus, for example, if $\rho = 1$, then $p(i) = 1/i(i+1)$. Several statistics, from species population to word frequencies, have been observed to obey a Yule-Simon distribution, most often with parameter $\rho = 1$ [2]. This particular distribution is also related to continued fractions, being the distribution of the first coefficient when the number being represented is chosen uniformly over the unit interval $(0, 1)$. For P_1^{YS}, Yokoo's code is 0.066% better than Code -1.

The estimates in Table II were calculated based on finite sums and estimates of the remaining infinite sum. For fixed codes and for entropy, these codes are as calculated in the Appendix, and are thus accurate to the precision given. The Golin code was estimated based on the partial code and conditional entropy of the remaining items. Similarly, optimal expected codeword lengths were estimated using an optimal code for the partial sum and the entropy of the remaining items; although not having the same guaranteed accuracy, the results seem to provide accurate estimates based upon the behavior of coding truncated probability distributions of increasing size. In [39], it is shown that sequences of such truncated distributions always have a subsequence converging to the optimal code, providing theoretical justification for the use of this technique. Values that are exactly calculated from infinite sums, rather than estimated, are indicated by the reduced number of figures (for multiples of 0.1) or through ellipses in the case of

$$2.66666\ldots = \frac{5}{3}, 1.94737\ldots = \frac{\zeta(1.5)}{\zeta(2.5)}, \text{ and } 1.36843\ldots = \frac{\zeta(2)}{\zeta(3)}.$$

These values are exactly known due to being means of Yule-Simon and zeta distributions, which are known in closed form. In addition, the average length of the Elias γ code (EG0) code for a Yule-Simon distribution with $\rho = 1$ is easily calculated as

$$\begin{aligned}\sum_{i=1}^{\infty} p(i)n(i) &= 1 + 2\sum_{i=1}^{\infty}\frac{\lfloor \lg i \rfloor}{i(i+1)} \\ &= 1 + 2\sum_{j=0}^{\infty} j \sum_{i=2^j}^{2^{j+1}-1}\left(\frac{1}{i} - \frac{1}{i+1}\right) \\ &= 1 + 2\sum_{j=0}^{\infty} j 2^{-j-1} = 3.\end{aligned}$$

Golin's algorithms both result in the same code for this distribution, since the algorithms' conditions result in groupings of probabilities summing to powers of two.

Excluding Golin codes, we find that the codes introduced here do quite well, only failing to improve upon existing fixed codes in one case, the Yule-Simon distribution with parameter $\rho = 1$ ($p(i) = 1/i(i+1)$). Because Yokoo's code requires computing codewords for complete binary codes with unequal codeword lengths, however, the codewords of codes introduced here require less computation to encode and decode. For all tested distributions, Yokoo's code and the codes introduced here are both strict improvements on exponential-Golomb and Elias codes, confirming that, in practice, strongly smooth codes are preferable to those lacking this property.

Note that not all known codes for integers were tested here; certain codes can be ruled out due to the length of the first few codewords (e.g., Even-Rodeh [40], Williams-Zobel [41]), whereas others lack the alphabetic property and/or have significantly higher computational complexity (e.g., Fibonacci [42], [43]). In comparison to feasible codes, the codes introduced here are a notable improvement. While not optimal, they can be quite useful in practical applications.

APPENDIX

Consider all codes and probability distributions that are monotonic and for which we can find $\alpha, \beta, \kappa > 0, \mu, \xi > 0, \tau > 0, \upsilon > 0, \phi > 0$ such that

$$n(i) \in [\tau \ln(i+\mu+1) + \alpha, \upsilon \ln(i+\mu) + \beta]$$

and

$$p(i) \in \left[\frac{\phi}{(i+\kappa)^{\xi+1}}, \frac{\phi}{i^{\xi+1}}\right]$$

for large enough $i \geq i_{\min}$. Then, for $x > i_{\min}$, we have

$$\begin{aligned}
\sum_{i=x}^{\infty} p(i)n(i) &\geq \int_x^{\infty} p(i)n(i-1)di \\
&\geq \int_x^{\infty} \frac{\tau\phi \ln(i+\mu) + \alpha\phi}{(i+\kappa)^{\xi+1}} di \\
&\geq \phi \int_x^{\infty} \frac{\tau \ln(i+\kappa) + \tau f_{\min}(x) + \alpha}{(i+\kappa)^{\xi+1}} di \\
&= \frac{\tau\phi \ln(x + \min(\kappa, \mu)) + \tau\phi\xi^{-1} + \alpha\phi}{\xi(x+\kappa)^{\xi}}
\end{aligned}$$

where $f_{\min}(x) = \min(\ln(x+\mu) - \ln(x+\kappa), 0)$, and

$$\begin{aligned}
\sum_{i=x}^{\infty} p(i)n(i) &\leq \int_x^{\infty} p(i-1)n(i)di \\
&\leq \int_x^{\infty} \frac{\upsilon\phi \ln(i+\mu) + \beta\phi}{(i-1)^{\xi}} di \\
&\leq \phi \int_x^{\infty} \frac{\upsilon \ln(i-1) + \upsilon f_{\max}(x) + \beta}{(i-1)^{\xi}} di \\
&= \frac{\upsilon\phi \ln(x + \max(-1, \mu)) + \upsilon\phi\xi^{-1} + \beta\phi}{\xi(x-1)^{\xi}}
\end{aligned}$$

where $f_{\max}(x) = \max(\ln(x+\mu)+\ln(x-1), 0)$, providing upper and lower bounds to average codeword length using code $N = \{n(i)\}$ for probability distribution $P = \{p(i)\}$. Other distributions (such as Golomb codes) and entropy can be bounded similarly.

Such an approach enables us to find estimates with accuracies limited only by the precision of the partial summations (i.e., round-off error). For the probability distributions currently under consideration, we have:

	ξ	ϕ	κ
P_{GK}	1	1	$\lg e$
P_{ρ}^{YS}	ρ	ρ	$\rho\Gamma(\rho+1)$
P_s^{ζ}	0	$s-1$	$\zeta^{-1}(s)$

In order to find bounds for expected codeword lengths, we should first define the codes we are using. Since we only care about codeword lengths, we use code definitions that apply to $\mathcal{X}$ and have the same lengths N as the (equivalent but possibly different) original definitions:

$$\begin{aligned}
\text{Elias } \gamma \quad c_\gamma(i) &= \begin{cases} 0, & i = 1 \\ 1c_\gamma\left(\frac{i}{2}\right)0, & i = \{2,4,6,\ldots\} \\ 1c_\gamma\left(\frac{i-1}{2}\right)1, & i = \{3,5,7,\ldots\} \end{cases} \\
\text{Elias } \delta \quad c_\delta(i) &= c_\gamma(1 + \lfloor \lg i \rfloor) b(i - 2^{\lfloor \lg i \rfloor}, 2^{\lfloor \lg i \rfloor}) \\
\text{Elias } \omega \quad c_\omega(i) &= \begin{cases} 0, & i = 1 \\ c'_\omega(\lfloor \lg i \rfloor) b(i) 0, & i > 1 \end{cases} \\
\text{JI} \quad c_{\text{JI}} &= \begin{cases} 0, & i = 1 \\ 1c_\omega(i-1), & i > 1 \end{cases} \\
\text{EG}k \quad c_{\text{EG}k}(i) &= c_\gamma\left(1 + \left\lfloor \tfrac{i-1}{2^k} \right\rfloor\right) b((i-1) \bmod 2^k, 2^k) \\
\text{Yokoo} \quad c_{\text{Yok}}(i) &= \begin{cases} 0, & i = 1 \\ 100, & i = 2 \\ 101, & i = 3 \\ 1^{g_i}00 b(i - 2^{g_i}, m_i), & i < q_i \\ 1^{g_i}01 b(i - q_i, 2^{g_i} - m_i), & i \geq q_i \end{cases}
\end{aligned}$$

where $c'_\omega(i)$ is all but the last bit of c_ω, $g_i \triangleq \lg i$, $m_i \triangleq (2^{g_i} - (-1)^{g_i})/3$, and $q_i \triangleq 2^{g_i} + m_i$. Recall that $b(j,k)$ denotes the $(j+1)$th codeword of a complete binary code with k items.

For these codes, $\alpha, \beta, \mu, \tau > 0, \upsilon > 0$ can be

	α	β	μ	τ	υ
γ, Yokoo	-1	1	0	$2\lg e$	$2\lg e$
JI ($i > 1$)	2	2	-1	$\lg e$	$2.5\lg e$
Code k ($k \leq 0$) ($i > -k$)	$\alpha_0 - k$	$-1-k$	$2+k$	$2\lg e$	$2\lg e$

where $\alpha_0 = 1 - 2\lg 3$. (Parameters for δ codes, ω codes, EGk codes, and Code k for $k > 0$ can be similarly formulated, but these are unused here, as the γ code is clearly better for all distributions considered.)

For finding the best code within code families with multiple codes — such as Code k, EGk, and Gk (Golomb code k, defined in the main text) — partial sums can be used to limit the number of codes tested to a finite number. For example, these codes have $n(1) \to \infty$ as $k \to +\infty$, so at some point $p(1)n(1)$ will be too large to consider Code k with parameters $k > k_{\max}$ for some $k_{\max}$. Similarly, as $k \to -\infty$, the unary portion of the code can be used for the partial sum.

Lacking $\alpha, \beta, \mu, \tau, \upsilon$, an obvious lower bound for $\sum_{i=1}^{\infty} p(i)n(i)$ is $\sum_{i=1}^{x-1} p(i)n(i)$, but a much more accurate bound can be found via entropy bounding with a value of x such that $\sum_{i=1}^{x-1} 2^{-n(i)} = 1 - 2^{-y_x}$ for some y_x. For such values, since the code can be assumed without loss of generality to be monotonic, the codewords can be assumed to be all the leaves of a subtree rooted at depth y_x. Since any normalized tree is subject to the entropy bound $\sum_i p(i)n(i) \geq H(P)$, we can normalize to find a useful bound for the overall code. Let us first assign

$$\begin{aligned}
\sigma_x &\triangleq \sum_{i=1}^{x-1} p(i), \qquad H_x \triangleq \sum_{i=x}^{\infty} p(i) \lg \frac{1}{p(i)} \\
H_x^{\text{cond}} &\triangleq \sum_{i=x}^{\infty} \frac{p(i)}{1-\sigma_x} \lg \frac{1-\sigma_x}{p(i)} = \lg(1-\sigma_x) + \frac{H_x}{1-\sigma_x}
\end{aligned}$$

where H_x can be lower-bounded by as previously described. Thus, applying the entropy bound to the normalized subtree,

$$\sum_{i=x}^{\infty} \frac{p(i)}{\sum_{j=x}^{\infty} p(j)} (n(i) - y_x) \geq H_x^{\text{cond}}$$

so

$$\begin{aligned}
\sum_{i=1}^{\infty} p(i)n(i) &\geq (y_x + H_x^{\text{cond}})(1-\sigma_x) \\
&= H_x + (y_x + \lg(1-\sigma_x))(1-\sigma_x)
\end{aligned}$$

This is useful for the codes calculated on the fly, e.g., Golin's codes.

Golin's original approach, *alg1*, starts by finding the minimum value k_1 such that

$$\sum_{i=1}^{2^{k_1}} p(i) > \frac{3-\sqrt{5}}{2} = 0.381966\ldots$$

and assigning the first 2^{k_1} inputs code $0b(i-1, 2^{k_1})$. The algorithm then normalizes the remaining inputs and finds the minimum value k_2 such that

$$\sum_{i=2^{k_1}+1}^{2^{k_1}+2^{k_2}} p_1(i) > \frac{3-\sqrt{5}}{2} \text{ where } p_1(i) = \frac{p(i)}{1 - \sum_{j=1}^{2^{k_1}} p(j)}$$

and assigns the next 2^{k_2} inputs code $10b(i-1-2^{k_1}, 2^{k_2})$. Continuing as needed, the algorithm sequentially finds minimum k_h (given k_1 through k_{h-1}) such that

$$S(k_1^h, P) \triangleq \frac{\sum_{i=1+K(h-1)}^{K(h)} p(i)}{1 - \sum_{i=1}^{K(h-1)} p(i)} > \frac{3-\sqrt{5}}{2}$$

where $K(h) \triangleq \sum_{j=1}^{h} 2^{k_j}$, and assigns code

$$1^{h-1}0b\left(i - 1 - \sum_{j=1}^{h-1} 2^{k_j}\right)$$

to items $1+K(h-1) = 1+\sum_{j=1}^{h-1} 2^{k_j}$ through $K(h) = \sum_{j=1}^{h} 2^{k_j}$.

This top-down approach is quite similar to Shannon-Fano coding [44], a modification of which results in *alg2*, previously proposed in [45]. In this case, the the grouping condition is not the first k_h such that $S(k_1^h, P) > (3-\sqrt{5})/2$, but the k_h minimizing

$$|S(k_1^h, P) - 0.5|$$

that is, the group of a power of two that results in the most even division between those grouped and those left ungrouped. (Note that Shannon-Fano codes use the overall "best split" whereas these codes use the best split that groups items together in powers of two.)

REFERENCES

[1] V. Pareto, *Cours d'économie politique*. Geneva, Switzerland: Droz, 1896.

[2] H. A. Simon, "On a class of skew distribution functions," *Biometrika*, vol. 42, no. 3/4, pp. 425–440, 1955.

[3] F. Auerbach, "Das Gesetz der Bevölkerungskonzentration," *Petermanns Geographische Mitteilungen*, vol. 59, pp. 74–76, 1913.

[4] J. B. Estoup, *Gammes Stenographiques*. Paris, France: Institut Stenographique de France, 1916.

[5] G. K. Zipf, "Relative frequency as a determinant of phonetic change," *Harvard Studies in Classical Philology*, vol. 40, pp. 1–95, 1929.

[6] G. U. Yule, "A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, F.R.S." *Philos. Trans. Roy. Soc. London Ser. B*, vol. 213, pp. 21–87, 1925.

[7] C. F. Gauss, "Eine Aufgabe der Wahrscheinlichkeitsrechnung," 1800, in Werke Sammlung, Band 10 Abt 1, pp. 552–556, available from http://www-gdz.sub.uni-goettingen.de/cgi-bin/digbib.cgi?PPN235957348.

[8] R. O. Kuzmin, "Sur un problème de Gauss," in *Atti del Congresso Internazionale dei Matematici*, vol. 6, Sept. 1928, pp. 83–89.

[9] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions," *Internet Mathematics*, vol. 1, no. 2, pp. 226–251, 2004.

[10] M. E. J. Newman, "Power laws, Pareto distributions and Zipf's law," *Contemporary Physics*, vol. 46, no. 5, pp. 323–351, Sept. 2005.

[11] N. N. Taleb, *The Black Swan: The Impact of the Highly Improbable*. New York, NY: Random House, 2007.

[12] J. Abrahams, "Code and parse trees for lossless source encoding," *Communications in Information and Systems*, vol. 1, no. 2, pp. 113–146, Apr. 2001.

[13] J. Teuhola, "A compression method for clustered bit-vectors," *Inf. Processing Letters*, vol. 7, no. 6, pp. 308–311, Oct. 1978.

[14] P. Elias, "Universal codeword sets and representations of the integers," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 2, pp. 194–203, Mar. 1975.

[15] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, July 2003.

[16] D. Marpe, G. Blättermann, and T. Wiegand, "Improved cabac," in *ITU-T Q.6/SG16, VCEG-O18*, Dec. 4–6, 2001, available from http://ftp3.itu.ch/av-arch/jvt-site/2001_12_Pattaya/VCEG-O18.doc.

[17] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[18] D. E. Knuth, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 3rd ed. Reading, MA: Addison-Wesley, 1997.

[19] D. W. Matula and P. Kornerup, "An order preserving finite binary encoding of the rationals," in *Proc., 6th Symposium on Computer Arithmetic*, 1983, pp. 201–209.

[20] H. Yokoo, "An efficient representation of the integers for the distribution of partial quotients over the continued fractions," *J. Inform. Processing*, vol. 11, no. 4, pp. 288–293, 1988.

[21] S. W. Golomb, "A class of probability distributions on the integers," *Journal of Number Theory*, vol. 2, no. 2, pp. 189–192, May 1970.

[22] A. Kato, "Huffman-like optimal codes and search codes for infinite alphabets," 1997, unpublished manuscript.

[23] S. W. Golomb, "Run-length encodings," *IEEE Trans. Inf. Theory*, vol. IT-12, no. 3, pp. 399–401, July 1966.

[24] R. G. Gallager and D. C. van Voorhis, "Optimal source codes for geometrically distributed integer alphabets," *IEEE Trans. Inf. Theory*, vol. IT-21, no. 2, pp. 228–230, Mar. 1975.

[25] J. Abrahams, "Huffman-type codes for infinite source distributions," *Journal of the Franklin Institute*, vol. 331B, no. 3, pp. 265–271, May 1994.

[26] T. Chow and M. Golin, "Convergence and construction of minimal-cost infinite trees," in *Proc., 1998 IEEE Int. Symp. on Information Theory*, Aug. 1998, p. 227.

[27] N. Merhav, G. Seroussi, and M. Weinberger, "Optimal prefix codes for sources with two-sided geometric distributions," *IEEE Trans. Inf. Theory*, vol. IT-46, no. 2, pp. 121–135, Mar. 2000.

[28] M. J. Golin and K. K. Ma, "Algorithms for constructing infinite Huffman codes," Hong Kong University of Science & Technology Theoretical Computer Science Center, Tech. Rep. HKUST-TCSC-2004-07, Aug. 2004, available from http://www.cs.ust.hk/tcsc/RR/index_7.html.

[29] F. Bassino, J. Clément, G. Seroussi, and A. Viola, "Optimal prefix codes for two-dimensional geometric distributions," in *Proc., IEEE Data Compression Conf.*, Mar. 28–30, 2006, pp. 113–122.

[30] M. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Image Processing*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000, originally as Hewlett-Packard Laboratories Technical Report No. HPL-98-193R1, November 1998, revised October 1999. Available from http://www.hpl.hp.com/loco/.

[31] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sept. 1952.

[32] J. L. Bentley and A. C. Yao, "An almost optimal solution for unbounded searching," *Inf. Processing Letters*, vol. 5, no. 3, pp. 82–87, Aug. 1976.

[33] R. Ahlswede, T. S. Han, and K. Kobayashi, "Universal coding of integers and unbounded search trees," *IEEE Trans. Inf. Theory*, vol. IT-43, no. 2, pp. 669–682, Mar. 1997.

[34] P. A. Humblet, "Optimal source coding for a class of integer alphabets," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 1, pp. 110–112, Jan. 1978.

[35] S. J. Golin, "A simple variable-length code," *Signal Processing*, vol. 45, no. 1, pp. 23–35, Mar. 1995.

[36] V. I. Levenshtein, "On the redundancy and delay of separable codes for the natural numbers (об избыточности и замедлении разделимого кодирования натуральных чисел)," *Problems of Cybernetics*, vol. 20, pp. 173–179, 1968.

[37] J. Wen and J. D. Villasenor, "Structured prefix codes for quantized low-shape-parameter generalized Gaussian sources," *IEEE Trans. Inf. Theory*, vol. IT-45, no. 4, pp. 1307–1314, May 1999.

[38] P. Kornerup and D. W. Matula, "LCF: A lexicographic binary representation of the rationals," *J. Universal Comput. Sci.*, vol. 1, no. 7, pp. 484–503, July 1995.

[39] T. Linder, V. Tarokh, and K. Zeger, "Existence of optimal prefix codes for infinite source alphabets," *IEEE Trans. Inf. Theory*, vol. IT-43, no. 6, pp. 2026–2028, Nov. 1997.

[40] S. Even and M. Rodeh, "Economical encoding of commas between strings," *Commun. ACM*, vol. 21, no. 4, pp. 315–317, Apr. 1978.

[41] H. E. Williams and J. Zobel, "Compressing integers for fast file access," *The Comput. J.*, vol. 42, no. 3, pp. 193–201, 1999.

[42] R. M. Capocelli and A. De Santis, "Regular universal codeword sets," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 129–133, Jan. 1986.

[43] A. Apostolico and A. S. Fraenkel, "Fibonacci representation of strings of varying length using binary separators," *IEEE Trans. Inf. Theory*, vol. IT-33, no. 2, pp. 238–240, Mar. 1987.

[44] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, July 1948.

[45] L. R. Bahl and H. Kobayashi, "Image data compression by predictive coding II: Encoding algorithms," *IBM J. Res. Develop.*, vol. 18, no. 2, pp. 172–179, Mar. 1974.